Part 2.

1. Discuss the following terms: URL, URI and URN used in the context of web resources. Provide examples for each of them. (20%)

A URI is any characters string used to identify a web resource on the network by location or by name or by both. There are two types. Firstly, URLs: Uniform Resource Locator, this just refers to an address. This is the most common method used. Secondly, URNs: Uniform Resource Names. There needs to be consistency for finding Web resources, and this is the purpose of the uniformity of the URIs, URLs and URNs. These refer to a web resource such as a website.

A web resource should have a unique identifier, allowing it to be distinguished from other web resources or by uniquely addressable, giving it more individuality, or both.

All URNs and URLs are URIs, but not all URIs are URNs or URLs. URIs identify by name and location or both.

URNs are less effective as many names can be similar leading to overlap and this is why we URLs are more common for recognising web resources, as they are more unique.

The URL requires two sections. They are the scheme and the domain, also known as a host, or the IP address. URLs has a scheme at the beginning such as HTTPs. For example: <https://www.facebook.com/>. You can see the domain name here too: “Facebook”. The port is optional and should go after the domain/IP address. It’s a number used as a connection end point. If you don’t specify the Port Number, it has a default which it will just go with, which doesn’t appear on the URL. After this comes the path, again which is optional, and this points to a particular web resource in the domain you want. Next is the query String, this just provides more info to further specify which web resource you are looking for. It passes info from the client to the server. It starts with a question mark. It is always at the end of the URL. The values are separated by an ampersand.

Here’s a full example: [https://www.facebook.com/80/messages/friend’s&name](https://www.facebook.com/80/messages/friend's&name)

Explained: protocol//domain/port/path?Query&string

The URN is a location independent web resource identifier, instead it is identified by its unique name. This can proved clarity to a web resource such as an important document. The name will always remain the same.

URNs begin with a scheme specification. This is “urn”, this is followed by the NID, a namespace identifier. The scheme and the namespace are separated by a colon. The namespace is under the control of the Internet Assigned Numbers Authority (IANA). There are many different namespaces. We are going to use NID for our example. This is followed by a namespace specific string. The namespace identifier is separated from the namespace specific string by a colon. A

Here is an example of a URN: urn: NID: 9834234567223344

Explained: scheme: namespace identifier: namespace specific string

URIs, URNs and URLs are one of the wonders of the modern world. They have allowed us to easily access data across the internet by making the identification of Web resources unique whether it’s by name, location or both. This ease of naming Web Resources means efficiency, uniformity and is pragmatic.

1. HTTP messages are how data is exchanged between a server and a client on the web. Compare and contrast the following two types of messages: HTTP request and HTTP response. Provide one example for each type. (20%)

HTTP (Hypertext Transfer Protocol) is an application layer, client server protocol. It’s the basis of any data transfer on the Web, for example videos or documents. The client and the servers communicate with each other by sending messages. The message is sent by the client: a request and received by a server: a response.

Usually the request comes from a Web Browser. It is identified by a given request URL, it’s case sensitive and should be upper case.

For the client and the server to communicate they must both use the HTTP protocol.

The HTTP request

The HTTP client must send a request to the server. This is done by a request method which specifies what instructions it needs carried out by the Web Resource.

Here are some of the most common methods:

The GET method gets the data using a given URI, it should obtain the data but not alter the data. This is one of the most common methods.

The HEAD method. This is the same as the GET Method but doesn’t have a message body in the response.

The POST method. This one of the most common methods. This is used to send data to the server to create or update.

The PUT method updates the resource.

The DELETE method, as its name indicates, deletes a resource specified in the URL.

The Options Method lets us know what the communications options are for a resource.

The Trace Method allows us to see what the received message looks like.

The Patch Method is like the PUT and the POST method but it’s only a partial modification to the resource.

The Request method is formed the following way. There is a request line. Followed by a header and then the body which is optional.

In the request line we have Verb, such as POST, coming first followed by a slash. Next comes the URI, the resource we want to access, then there is a space and then we have the type of HTTP version we want to use.

Here is an example of a request line: GET /menu.html HTTP/1.1

Explained: Verb / URI HTTP version

Next there are the headers. There can be many but I will use only three here.

Here is an example of the headers:

Host: www.funfactswebsite.com

Content-type: text:html

Accept-language: fr

The body is optional but you can include any information you feel you want to here.

factsId=42&year=Fastest+Gun+In+The+West

Here is the full example of Request Method:

GET /menu.html HTTP/1.1

Host: www.funfactswebsite.com

Content-type: text:html

Accept-language: fr

factsId=42&year=Fastest+Gun+In+The+West

The HTTP Response

Now that the HTTP Request is sent to the server, the HTTP Response goes to the client. The response method is a packet of data from the server to the client. It is as the name indicates a response to the request made previously to the client to the server. It send the data back if possible.

The response is made up of a Status Line followed by the headers, if there are any, and finally an optional message body.

In the Status Line, unlike the Request, there is no URI as that is only for the request. Instead we have the http/version followed by a status code and its associated text. The status code has three numbers, the first number define the response, and the other two digits don’t play a part in this definition. There are 5 classes of response and all of them have their own meaning.

100 - 199 = This means the order has been received and is being processed.

200 - 299 = This means the request was a complete success, it was accepted confirmed.

300 - 399 = This means it was redirected and there must be other action taken to complete the request.

400 - 499 = This means the request cannot be completed due to some error.

500 - 599 = This means the server has failed on its side.

Here’s an example of a status line – HTTP/1.1 200 OK

Next comes the Headers. The headers add more information to the Response.

For example:

Date: Sun, 30 Mar 2020 14:09:20 GMT\r\n

Server: GSE

Content-Type: text/html

Content-Length: 140

Connection: Closed

The body is separated from the Headers by a blank line. The message body contains the data the client requested. The message body is optional. For example here is the request for the homepage of a website.

<html>

<body>

<h1>Home Page</h1>

</body>

</html>

Here is the full example of the HTTP Request:

HTTP/1.1 200 OK

Date: Sun, 30 Mar 2020 14:09:20 GMT\r\n

Server: GSE

Content-Type: text/html

Content-Length: 140

Connection: Closed

<html>

<body>

<h1>Home Pageh1>

</body>

</html>

Now that we have talked about the differences between the Request and the Response, let’s sum up a comparison. The Request sends its instructions from the client to the server, while the Response Method sends a response to this Request from the Server back to the client. They are both based on HTTP.

The syntax is different but there are comparisons. The first line of both have similar names; The Request Line for the Request Method and the Status Line for the Response Method. The request Line has a verb at the start and the HTTP version at the end. The major difference is the URI in the middle which the Response doesn’t have. The status line, apart from the URI is quite similar. It has the HTTP version at the start however and the status code after which indicates the response, as well as the associated text clarifying the status code. This is similar to the verb in the Request Line.

The next part to both is the headers. This is very similar and indeed carries similar data. Finally both have an optional body at the end. The difference is in how this is laid out. It looks more like a URI in the request and front end code in the Response.

The Request and the Response Methods are fascinating communication methods which utilize HTTP protocol to request web resources from the web and the response to that request. The have similarities and differences. The can make sense even to the layman if he was to stop and read them. They are the basis for any data communication on the Web and we carry these out innumerable times per day without even noting it. Our lives would be vastly different without them.